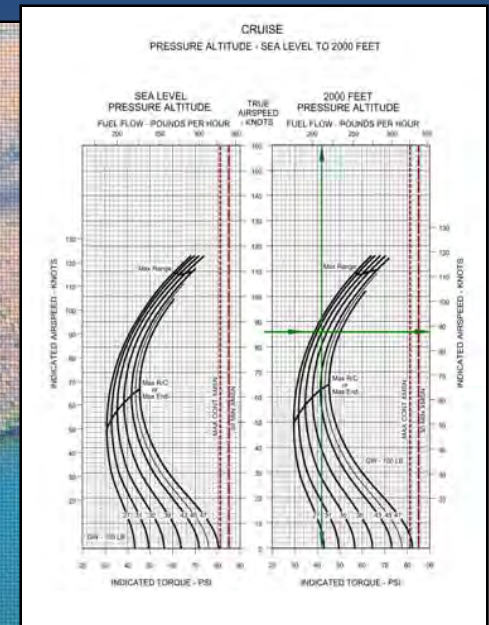


# CDI

Craft Designs, Inc.  
[www.craftdesigns.net](http://www.craftdesigns.net)

Software Engineering  
Specialist



## Software Test Appliance Techniques (STAT) for Software Systems

Ron Craft  
Craft Designs, Inc

May 2011

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE <b>MAY 2011</b>		2. REPORT TYPE		3. DATES COVERED <b>00-00-2011 to 00-00-2011</b>	
4. TITLE AND SUBTITLE <b>Software Test Appliance Techniques (STAT) for Software Systems</b>				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>Craft Designs, Inc,700 Boulevard South Suite 107,Huntsville,AL,35802</b>				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release; distribution unlimited</b>					
13. SUPPLEMENTARY NOTES <b>Presented at the 23rd Systems and Software Technology Conference (SSTC), 16-19 May 2011, Salt Lake City, UT. Sponsored in part by the USAF. U.S. Government or Federal Rights License</b>					
14. ABSTRACT <b>The dependencies in complex software systems are stretching industry software test capabilities such that schedules and budgets are constantly being compromised at the risk of producing software with more defects and reliability issues. STAT technologies are being integrating within our production code to facilitate improved testability and reliability. Modeled from techniques utilized in hardware systems commonly titled Built-in Test (BIT), STAT is used to develop applications that support testability without sched le compromising schedule and budget. The authors have found that STAT supports the development and deployment of robust software applications. ? Non-intrusive test techniques ? Improved reliability ? Supports test automation ? Improves integration success</b>					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT <b>Same as Report (SAR)</b>	18. NUMBER OF PAGES <b>32</b>	19a. NAME OF RESPONSIBLE PERSON
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>			

# Abstract

The dependencies in complex software systems are stretching industry software test capabilities such that schedules and budgets are constantly being compromised at the risk of producing software with more defects and reliability issues. STAT technologies are being integrating within our production code to facilitate improved testability and reliability. Modeled from techniques utilized in hardware systems commonly titled Built-in Test (BIT), STAT is used to develop applications that support testability without compromising schedule and budget. The authors have found that STAT supports the development and deployment of robust software applications.

- Non-intrusive test techniques
- Improved reliability
- Supports test automation
- Improves integration success

# Contents

The purpose of this briefing is to provide an understanding of what a Software Test Appliance is and how it can be applied to building robust and reliable Software Applications.

- Background
- Example Software Systems
- Non-intrusive test techniques
- Improved reliability
- Supports test automation
- Improves integration success

# Background

- 25+ years in Industry
- Emphasis on Software Engineering
- Embedded systems for Medical applications
- Embedded systems for Aircraft avionics.
- Numerical models for University Research
- High speed measurement tools

# Sample Software Systems

- Mission Planning Software
- Flight Performance Planning Models
- Embedded Flight Performance Models for Mission Execution



# Mission Planning and Performance Software

- Flight Performance Models - Digital models of aircraft performance flight capabilities and limitations
- Developed by combining flight-test and engineering data with standard mathematical models (equations) of aircraft performance
- Rotary Aircraft - Performance Planning Cards (PPC)
- Fixed Wing Aircraft – Take Off and Landing Data (TOLD)
- Desktop and Onboard Embedded Applications

TAKEOFF AND LANDING DATA CARD

File Edit Tools

ACFT: Aircraft Type  
#1 ETF: 0.9 #2 ETF: 0.97

TAKEOFF LANDING DATE: 11/19/2010  
Route Point 1 Route Point 2 ATF: 0.935

CAT: 0 0 °C  
PRESSURE ALTITUDE: 6000 6000 FT  
FLAT PLATE DRAG: 0 0 FT SQ  
TORQUE RATIO: 0.976309196240771 0.976309196240771 LBS  
OPERATING WEIGHT: 19000 17000 LBS  
LOAD (+): 23 63 LBS  
ZERO FUEL WEIGHT: 19025 17065 LBS  
MAIN FUEL: 45 10 LBS  
AUX FUEL: 15 16 LBS  
GROSS WEIGHT: 19065 17091 % Q  
MAX POWER AVAILABLE (2.5 MIN): 107.704121600662 107.704121600662 % Q  
(10 MIN): 106.593264955000 106.593264955000 % Q  
(30 MIN): 106.593264955000 106.593264955000 % Q  
POWER REQUIRED (OGE): 95.9252054069962 91.4735507912787 % Q  
(30): 81.0274014759630 69.5079691814477 % Q  
(10): 81.0274014759630 69.5079691814477 % Q  
MAX GROSS WEIGHT (OGE): 20483.076171875 20483.076171875 LBS  
(10): 22636.71875 22636.71875 LBS  
SINGLE ENGINE AIRSPEED  
MINIMUM / MAXIMUM: 41.19874 / 94.53993 27.06904 / 104.1210 KIAS  
DUAL ENGINE A/S and TRQ  
MAXIMUM RANGE: 67.52285 / 47.67925 64.44161 / 42.12295  
MAX END: 118.8293 / 78.15065 120.1639 / 66.83443  
Vh: 137.5924 141.92439 KIAS  
VNE: 166.935833000961 182.141300946305 KIAS  
(MAX ANGLE OF BANK)  
GROSS WT PA CAT AIRSPEED MAX BANK ANGLE FOR BLADE STALL  
20000 6000 40 90 38.438232421875  
10000 7500 0 140 23.4375

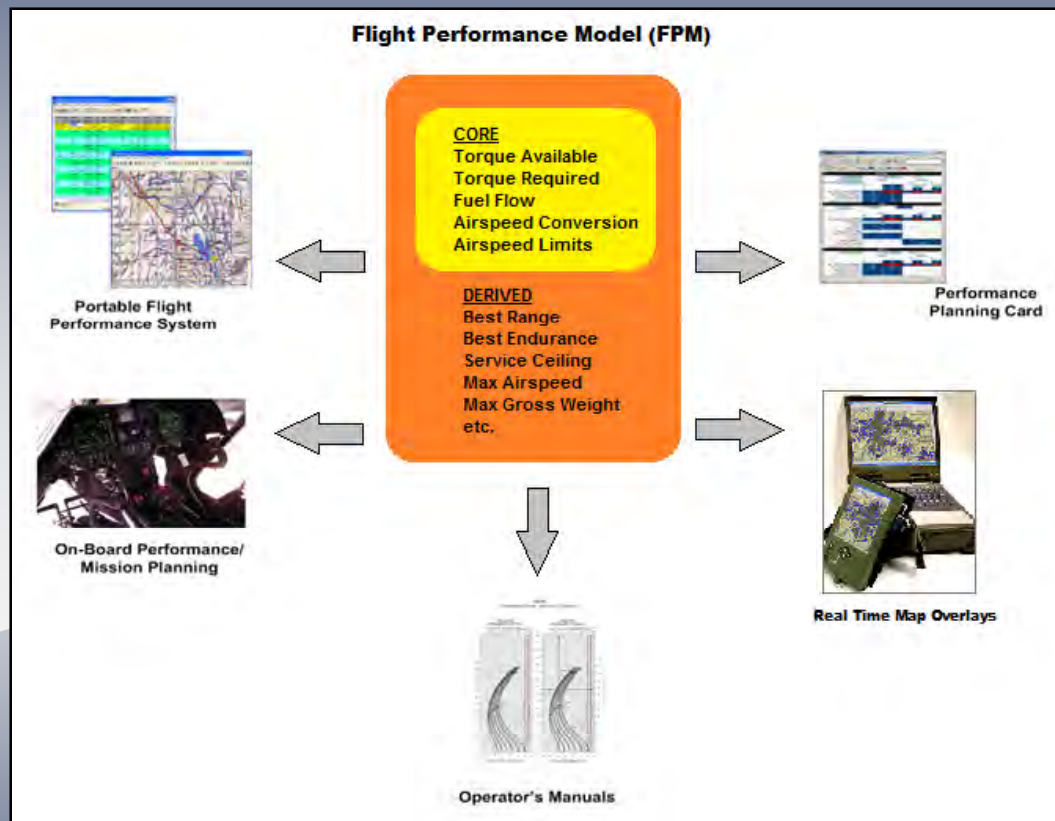
CALCULATE  
ADDITIONAL INPUTS  
DRAG FACTOR

## – Platforms

- Apache
- Blackhawk
- Chinook
- Kiowa warrior
- JCA
- Sherpa
- Citation



# Flight Performance Models - Desktop



– Flight Performance Model integrated with desktop applications

- Integrated Performance and Aircraft Configuration (IPAC)
- Portable Flight Planning Software (PFPS)
- Aviation/Joint Mission Planning System (AMPS/JMPS)
- Falconview
- Operators Manual Charts



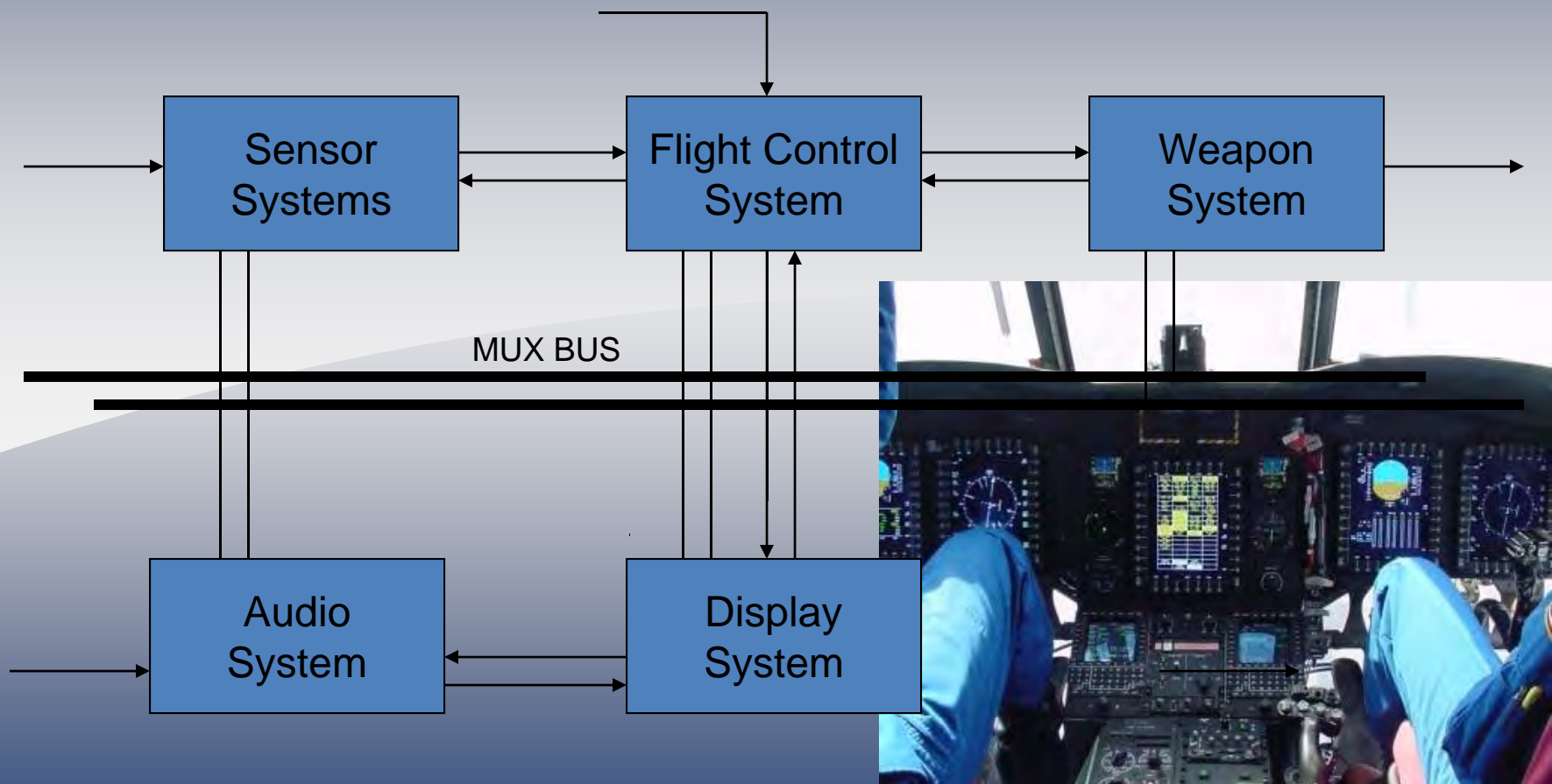
# Flight Performance Models - Embedded

- Embedded Flight Performance Models (EFPM) are being inserted into fixed wing and rotary wing cockpits
- Enables onboard flight performance and mission planning
- Accurate and consistent with desktop applications
- Very fast execution, very small memory requirements
- Efficient numerical methods to create faster EFPMs
  - Non-dimensional data
  - Higher-order interpolation
  - Pre-processing
- Currently have EFPM onboard
  - OH-58D
  - UH-60M
  - CH-47F
  - MH-47G
  - MH-60K/L/M



# Systems within a System

Today's systems are becoming systems of Software Systems



# Software Systems Components

- Standardized interfaces
- Abstracted capabilities
- Functions
  - Producer
  - Consumer
  - Mediators
- Coupling Complexity
  - Data
  - Control

# Data Coupling

- Data coupling - The dependence of a software component on data not exclusively under the control of that software component. (DO-178B)
  - Define the behavior for the input domain
  - Bounds check the component
    - Outside the bounds
    - At the bounds
    - The entire domain

# Control Coupling

- Control coupling - The manner or degree by which one software component influences the execution of another software component.  
(DO-178B)
  - Requirements must fully specify switches
  - Test must exercise each switch
  - Reduce control coupling, reduce test cases

# Goals of Testing

- Prevent bugs in software and hardware before deployment.
- Discover symptoms of bugs before they affect safety or functionality of systems.
- Provide diagnostic information on detected bugs.



# Goals of Software Test Appliance

Goals of testing listed above plus:

- Non-intrusive verification of proper functionality of systems for operational system status.
- Non-intrusive data collection to support the verification of required operation.
- Detection of changes in SW configurations and operation during power up and normal operations.
- Provide a mechanism to detect changes in data
- Provide a mechanism to support developmental unit test
- Provide a mechanism to support verification, validation, and qualification test

# Types of Software Test Appliance

Techniques designed for hardware systems can be adapted for software systems

- Software Built-in Test (SW-BIT)
- Interface Logger (SW-INF)

# Modes of SW-BIT

- Startup BIT
  - Evaluation of key functions and capabilities before transitioning to operational system status.
- Continuous BIT
  - Evaluation of selected capabilities during operational system status.

# Modes of BIT (Cont.)

- Initiated BIT
  - Detailed BIT used to provide diagnostic information while temporarily transitioned to non-operational system status.
- Maintenance BIT
  - Exhaustive BIT designed to operate with a maintenance interface and provide “peek and poke” capabilities into system during both operational and non-operational system status.

# Startup BIT

- Usually performed by the Boot Loader software.
- Evaluates memory locations using Destructive Stuck-on-1/Stuck-on-0 tests (memory is erased before download operations start).
- Downloads and verifies the operational code using sequence checking and check summing of the operational code.
- Activates all interfaces and verifies that they are operational by receiving/sending heartbeat messages.
- Activates operational code and verifies when it is running.

# Continuous BIT (CBIT)

- Foreground Tests:
  - Inputs:
    - Checksum, parity checks, time tags, sequence numbers, and heartbeat checks of digital and discrete inputs.
    - Voltage, current, frequency checks of analog and power inputs.
  - Processors
  - Software
- Background Tests:
  - Inputs:
    - Perform loopback tests of digital, discrete, and analog input.
    - Non-destructive Stuck-on-1/Stuck-on-0 tests on interface buffers.
  - Memory:
    - Non-destructive Stuck-on-1/Stuck-on-0 tests of all memory locations.



# Initiated BIT (IBIT)

- Detailed evaluations that may replace Startup BIT when adequate startup testing is too time-consuming.
- May be performed by operational code, however, IBIT is not performed during normal operation.
- Supports maintenance by:
  - Identifying *where* problems exist as well as problem types.
  - Providing an interface for maintenance software to access memory locations, etc.
  - Performing download evaluations.

# Maintenance BIT

- Development Platform:
  - Provides access to selected memory locations, by setting of breakpoints, etc., used to evaluate the software and/or hardware.
  - Sets up emulated/simulated inputs and stimuli.
- Repair Operations:
  - Downloads new software via maintenance interfaces.
  - Identifies sources of problems for repair operations on LRU/SRUs.
  - Evaluates repair status.

# Typical Test Procedures - Software

- Operational Evaluations Only – Does not include startup, development and V&V evaluations/tests.
- Examples of Operational Evaluations:
  - Data Analysis:
    - Perform sanity checks on input data.
    - Prevent run-time errors by insuring incorrect and out-of-bounds data are not used.
  - Stack Overflow – Provide software checks to insure against and report conditions where stacks overflow (especially necessary in “C”, C++, and other languages).
  - Exception Handling – Provide exception handling capabilities in the code development.

# SW-BIT: Detecting changes in SW configurations

- Startup SW-BIT checks:
  - Presence of data sources
  - Integrity of data sources (CRC)
  - Use of correct SW computational components (compare computed results against pre-computed expected results)
  - Expected behaviors from selected functional SW components
  - Correct model instantiation (software components, unique parameters, and specific data sources)

# SW-BIT: Detecting changes during operation

- Continuous SW-BIT checks:
  - Integrity of data sources (CRC)
  - Use of calculation status (NaN & status flags)
  - Memory leaks
  - Buffer overruns
  - Program flow by choosing test cases that will to maximize code coverage

## SW-BIT: During non-operation status

- Initiated SW-BIT supports regression testing:
  - SW tests that run during startup and under normal operation still return expected results
  - Test case stimulus chosen to maximize code coverage
- Maintenance SW-BIT
  - Test stubs
  - Upgrade verification/status



# SW-BIT Summary

- Checks for expected:
  - Computational results
  - Control Flow
  - Required behaviors
  - Hardware and software system configurations
- Flags non-expected results
- Supports developmental test
- Logs test case stimulus for analysis

# Interface Logging

- Include in operational requirements
- Test completely to avoid false failures
- Save all input information
- Save all output information
- Save needed state information
- Strategically capture the call trace
- Develop parsing tools to support analysis
- Identify interfaces where data can easily be gathered without intrusion.

# Technique to enable logging

- Pick a simple technique that will not be accidentally enabled

```
//  
// See if Test Appliance is ready to be enabled  
//  
    //does the user want to enable logging?  
    fopen_s(&logfile, "c:\\{5432testapp-papalog123}    \  
                \\eg45fhtymightbeagoodname.txt", "r");  
    if (logfile != NULL)  
    {  
        logging = true;  
        fclose(logfile);  
    }
```

# Logging Appliance

```
//  
// Did we find a mode to calculate  
//  
if(i < pModel->cModes)  
{  
    if (logging == true )  
    {  
        // Write State info and input info.  
        // Flush the file  
        // Close the File  
    }  
  
    ierr = pMyMode->prep_and_calc();    // Calculate the mode  
  
    if (logging == true)  
    {  
        // Write the output info and state info.  
        // flush the file  
        // close the file  
    }  
}
```

# Parser Output

**Log Parser**

Log File: C:\mylogh

Inputs

- 421 Anti-Ice: 0.000000
- 1714 Blade Erosion Kit: 0.000000
- 8269 Cabin Doors: 0.000000
- 8279 Cockpit Doors: 0.000000
- 1563 CONFIGURATION: 0.000000
- 8280 Crew Chief/Gunner Windows: 0
- 8267 ECS: 0.000000
- 8270 ECS State: 0.000000
- 8263 EIBF: 0.000000
- 8264 EIBF Bypass Doors: 0.000000
- 1772 Engine #1 Torque Factor: 1.000000
- 1773 Engine #2 Torque Factor: 1.000000
- 8383 Engine Type: 1.000000
- 8273 Fixed/Additional Download Cha
- 8756 Fixed/Additional Drag: 0.000000
- 1138 Free Air Temperature: 35.000000
- 744 Heater: 0.000000
- 1295 IGE Hover Height: 10.000000
- 8272 Internal/Additional Load Weight
- 8265 IR Suppression: 1.000000
- 8274 Jettisonable Stores Download C
- 8023 Jettisonable Stores Drag: 0.000000
- 736 Jettisonable Stores Weight: 0.000000
- 8288 Max Structural Weight: 22000.0
- 8268 OBOGS: 0.000000
- 8271 OBOGS State: 0.000000
- 1715 Operating Limit - DE: 2.000000
- 1560 Operating Limit - SE: 3.000000
- 1955 Operating Weight: 12000.000000

**Log Parser**

Log File: C:\mylogtrigger\sampleLogFile.txt

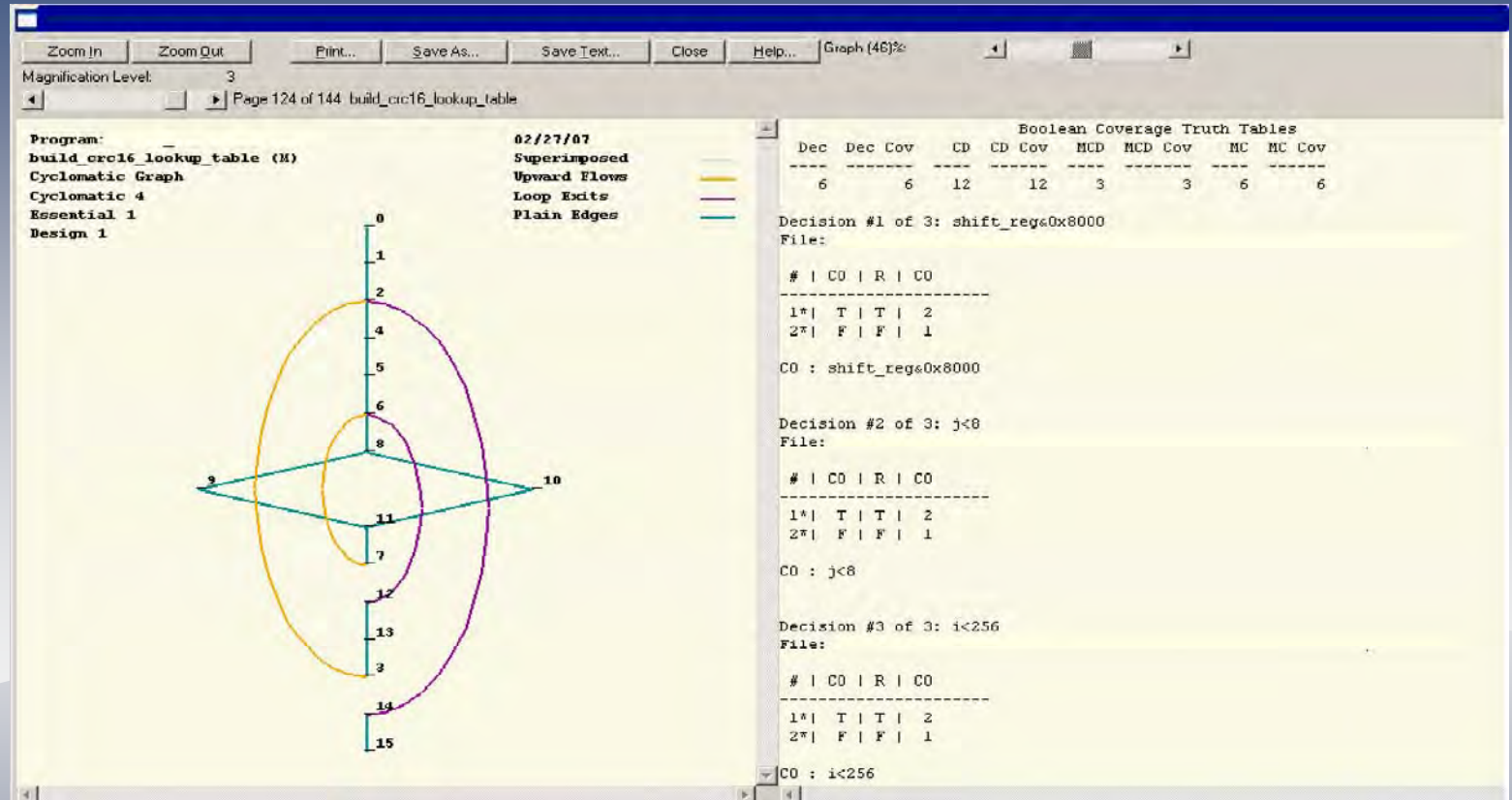
Delete Log Parse File

Enable Logging

Outputs

- CORE::Min\_Airspeed: IP: 1108
- Input Array
- Accessories Array
- Core Values
- Core Statuses
- CORE::Convert\_Airspeed:
- CORE::Min\_Airspeed:
- CORE::Convert\_Airspeed:
- 730 Gross Weight: 14412.000000
- 855 MAX ALLOWABLE GWT IGE: 22000.0
- 856 MAX ALLOWABLE GWT OGE: 20331.0
- 1718 MAX HOVER HEIGHT: 1000.000000
- 1785 MAX TORQUE AVAILABLE (Dual): 1000.000000
- 1788 MAX TORQUE AVAILABLE (Single #1): 1000.000000
- 1789 MAX TORQUE AVAILABLE (Single #2): 1000.000000
- 8307 MIN SE - IAS - W/O STORES: 13.000000
- 8308 MIN SE - IAS - W/STORES: 13.000000
- 1324 PREDICTED HOVER TORQUE (DE): 1000.000000
- 1804 PREDICTED HOVER TORQUE (SE#1): 1000.000000
- 1805 PREDICTED HOVER TORQUE (SE#2): 1000.000000
- 1790 Total Download Change: 0.000000
- 581 Total Flat Plate Drag: 0.000000
- 1180 TR (Dual): 1.000000
- 1791 TR (Single #1): 1.000000
- 1792 TR (Single #2): 1.000000
- Status: 0
- Messages
- IPAC Arrival
- Torque Required (111.0%) exceeds Torque
- JD\$LIST 1804
- IPAC Arrival

# McCabe as a Test Appliance



Note: 3 decisions, 6 possible outcomes



# Interface Logging Summary

- Provides call trace
- Parser can flag non-expected results
- Supports developmental test
- Logs test case stimulus for analysis
- Captured data can be fed back into application as a stimulus or regression
- Provides tangible test results